Tracing

Bug 1: Incorrect calculation of fines.

The first method call is on the Main.java file. When the user selects option R for returning book.

```
case "R":
    returnBook();
    libraryHelper.saveLibrary(library);
    break;
```

The returnBook method initilializes the returnBookControl class.

```
private static void returnBook() {
    IReturnBookControl returnBookControl = new ReturnBookControl(library);
    new ReturnBookUI(returnBookControl).run();
}
```

In the returnBookControl class the library calls the calculateOverDueFine method to calculate the overdue fine.

```
if (currentLoan.isOverDue()) {
    overDueFine = library.calculateOverDueFine(currentLoan);
    currentLoan.getPatron().incurFine(overDueFine);
}
```

The calculateOverDueFine method in the library class references the calendar class as shown below.

```
@Override
public double calculateOverDueFine(ILoan loan) {
    double fine = 0.0;
    if (loan.isOverDue()) {
        Date dueDate = loan.getDueDate();
        long daysOverDue = Calendar.getInstance().getDaysDifference(dueDate);
        fine = daysOverDue * FINE_PER_DAY;
    }
    return fine;
}
```

If the program is ran with a loan overdue with 1 day, the getDaysDifference method returns 0 instead of 1.

```
Choice : r
Return Book Use Case UI
28/10/2020

Scan Book (<enter> completes): 2
0
Inspecting
Book: 2
  Title:  celtics
  Author: rajon rondo
  CallNo: 123
  State:  ON_LOAN
Loan:  17
  Borrower 1 : Doe, John
  Book 2 : celtics
  DueDate: 27/10/2020
  State: OVER_DUE
```

From the screenshot above, the code is modified to display the days overdue when the date is 28 and the due date is 27. It should return 1 but it returns 0.
So the bugs originates from the calculation of days which happens in the method below;

```
@Override
public synchronized long getDaysDifference(Date targetDate) {
    long diffMilliseconds = getDate().getTime() - targetDate.getTime();
    long diffDays = diffMilliseconds / MILLIS_PER_DAY;
    return diffDays;
}
```

Bug 2: Incorrect fine amount levied.

The first method call is on the Main.java file. When the user selects option R for returning book.

```
case "R":
    returnBook();
    libraryHelper.saveLibrary(library);
    break;
```

The returnBook method initilializes the returnBookControl class.

```
private static void returnBook() {
    IReturnBookControl returnBookControl = new ReturnBookControl(library);
    new ReturnBookUI(returnBookControl).run();
}
```

In the returnBookControl class the library calls the calculateOverDueFine method to calculate the overdue fine.

```
if (currentLoan.isOverDue()) {
    overDueFine = library.calculateOverDueFine(currentLoan);
    currentLoan.getPatron().incurFine(overDueFine);
}
```

The calculateOverDueFine method in the library class references the calendar class as shown below.

```
@Override
public double calculateOverDueFine(ILoan loan) {
    double fine = 0.0;
    if (loan.isOverDue()) {
        Date dueDate = loan.getDueDate();
        long daysOverDue = Calendar.getInstance().getDaysDifference(dueDate);
        fine = daysOverDue * FINE_PER_DAY;
    }
    return fine;
}
```

By tracing the calculateDays method and adding a line to show the number of days overdue when the date is 28 and the overdue date is 22, the method returns 3 instead of 6 as shown below.

```
Choice : r
Return Book Use Case UI
28/10/2020

Scan Book (<enter> completes): 2
days overdue: 3
Inspecting
Book: 2
  Title:  celtics
  Author: rajon rondo
  CallNo: 123
  State:  ON_LOAN
Loan:  18
  Borrower 1 : Doe, John
  Book 2 : celtics
  DueDate: 22/10/2020
  State: OVER_DUE

Overdue fine : $3.00
Is book damaged? (Y/N):
```

From the trace we can see that the problem is in the days overdue calculation.

Bug 3: Incorrect state of a loan

The state of a loan is updated when the date is changed. From the main menu, when the user selected option t as shown below.

```
private static void incrementDate() {
    try {
        String daysString = getUserInput( prompt: "Enter number of days: ");
        int days = Integer.valueOf(daysString).intValue();
        calendar.incrementDate(days);
        library.checkCurrentLoansOverDue();
        Date currentDate = calendar.getDate();
        String dateString = dateFormat.format(currentDate);
        output(dateString);

    } catch (NumberFormatException e) {
        output( object: "\nInvalid number of days\n");
    }
}
```

From this method the checkCurrentLoansOverDue method in the library class is called.

```
@Override
public void checkCurrentLoansOverDue() {
    Date currentDate = Calendar.getInstance().getDate();
    for (ILoan loan : currentLoans.values()) {
        loan.updateOverDueStatus(currentDate);
        if (loan.isOverDue()) {
            IPatron patron = loan.getPatron();
            patron.restrictBorrowing();
        }
    }
}
```

In the checkCurrentLoansOverDue method the uploadOverDueStatus method in the loan class is called.

```
@Override
public void updateOverDueStatus(Date currentDate) {
    if (state == LoanState.CURRENT && currentDate.after(dueDate)) {
        this.state = LoanState.OVER_DUE;
    }
}
```

Thus from this trace we can conclude the bug is occurring in the method above.

Bug 4: Incorrect patron status

When the user selects the pay fine option in the main menu the payFineControl class is initialized.

```
private static void payFine() {
    IPayFineControl payFineControl = new PayFineControl(library);
    new PayFineUI(payFineControl).run();
}
```

In the payFineControl class the payFine method in the library class is called.

```
@Override
public double payFine(double amount) {
    if (!state.equals(ControlState.PAYING)) {
        throw new RuntimeException("PayFineControl: cannot call payFine except in PAYING state");
    }
    double change = library.payFine(patron, amount);
    if (change > 0) {
        String changeDisplayString = String.format("Change: $%.2f", change);
        payFineUi.display(changeDisplayString);
    }
    payFineUi.display(patron);
    payFineUi.setState(IPayFineUI.UIStateConstants.COMPLETED);
    state = ControlState.COMPLETED;
    return change;
}
```

In the payfine method in the library class, the payfine method in the patron class is called.

```
@Override
public double payFine(IPatron patron, double amount) {
    double change = patron.payFine(amount);
    setPatronBorrowingRestrictions(patron);
    return change;
}
```

In the method above, setPatronBorrowingRestrictions is set which is where the bug is coming from.

```
private void setPatronBorrowingRestrictions(IPatron patron) {
    if ((patron.getNumberOfCurrentLoans() >= LOAN_LIMIT) ||
            (patron.getFinesPayable() > MAX_FINES_OWED) ||
             patron.hasOverDueLoans()) {

        patron.restrictBorrowing();
    }
    else {
        patron.allowBorrowing();
    }
}
```

The method above references the ILibary interface where the MAX_FINES_OWED variable is set.

```
public interface ILibrary {

    static final int LOAN_LIMIT = 2;
    static final int LOAN_PERIOD = 2;
    static final double FINE_PER_DAY = 1.0;
    static final double MAX_FINES_OWED = 1.0;
    static final double DAMAGE_FEE = 2.0;
```

Bug 5: Negative fine amount

The fine is updated when the date is changed. From the main menu, when the user selected option t as shown below.

```
private static void incrementDate() {
    try {
        String daysString = getUserInput( prompt: "Enter number of days: ");
        int days = Integer.valueOf(daysString).intValue();
        calendar.incrementDate(days);
        library.checkCurrentLoansOverDue();
        Date currentDate = calendar.getDate();
        String dateString = dateFormat.format(currentDate);
        output(dateString);

    } catch (NumberFormatException e) {
        output( object: "\nInvalid number of days\n");
    }
```

From this method the checkCurrentLoansOverDue method in the library class is called.

```java
@Override
public void checkCurrentLoansOverDue() {
    Date currentDate = Calendar.getInstance().getDate();
    for (ILoan loan : currentLoans.values()) {
        loan.updateOverDueStatus(currentDate);
        if (loan.isOverDue()) {
            IPatron patron = loan.getPatron();
            patron.restrictBorrowing();
        }
    }
}
```

In the checkCurrentLoansOverDue method the uploadOverDueStatus method in the loan class is called.

```java
@Override
public void updateOverDueStatus(Date currentDate) {
    if (state == LoanState.CURRENT && currentDate.after(dueDate)) {
        this.state = LoanState.OVER_DUE;
    }
}
```

Thus from this trace we can conclude the bug is occurring in the method above.